



OpenOffice UX development.

Fluid Feel

November 29, 2009

Summary

The “Fluid Feel” command management idea is aimed at taking the current prototype design for command management, and making it more efficient and friendly. The final fusion of ideas from the development group will (hopefully) be a system that is not only efficient and pleasurable to use, but one that outstrips that of other Office Suites, instead of lagging in their wake. This new system will also prepare Openoffice for increased usage of touch-screen devices, with the system’s simple and straightforward, uncluttered design. The result will be, in conclusion, something that follows many of the goals of the renaissance project: making the user *desire* to use OOO, making a system where there is almost no wasted space, and, finally, allowing the user to experiment with their Office Suite, and discover something easier every time.

Contents:

- **Summary**
- **Mockup: Main Menu**
 - Explanation
 - Reasoning
- **Mockup: Color Picker**
- **Mockup: Font/Text picker**
- **Mockup: Stroke/Line picker**
- **Conclusion**

Mockup: Main Menu



This is positioned in a similar manner relative to the page as is MSOffice's Ribbon. This one is set up for Impress.

The default set up for the Main Menu (If someone can think of a catchy name, let me know) in Impress only shows these four categories.

!!!!Important!!!! The Menu's commands are all about modifying the document, i.e., adding objects, animating it, formatting it, etc. Commands that have to do with changing the way you work with the document, i.e., view, spellcheck, etc., remain in the menubar (File, Edit, Etc.).

Now, as you can see above, the bar is divided into 4 clickable regions. Each one will pulsate/glow blue when the mouse hovers over. If we want this to be an animation, or just a one-step process, we can do that. This theme, by the way, is entirely my own, as are the icons, and any official generation will have to be in OOO style, using the Galaxy icons.

Anyway, let's get to what happens when the user clicks on one of the regions, say, Format.

The other, non-selected regions "scoot" out of the way, and the selected region expands, to encompass a table of

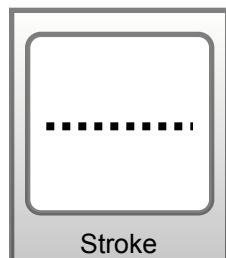


commands. Each of these commands glow in a way that reflects the glowing of the major groupings.

There are two types of these commands, simple and complex. A simple one, say, stroke/line, will look like this:

In the center of the command, there is a large (white) region which contains a thumbnail or preview. In the case of

"Stroke", it is quite possible to have a preview of the line that will stroke the object. In other cases, a thumbnail image representing the command will suffice. Below the thumbnail is the label, providing exact identification of the command. **The entire region is a button.**



In the case of a **complex** command (Right), only the larger region that encompasses the thumbnail is a button. A complex command occurs when there are several mutually exclusive command choices, i.e., the choice to fill with a color, gradient, or image. Then the user can choose the type from the choices on the left of the command, and click on the right area. Next, we move on to **clicking the button.**



Mockup: Color Picker

When the user clicks on a command, in this case, the “Color Fill” command, a small, semi-opaque window will appear. This window also needs a catchy name. Anyway, this window is probably the most important part of this proposal. In this case, we have a color picker window.

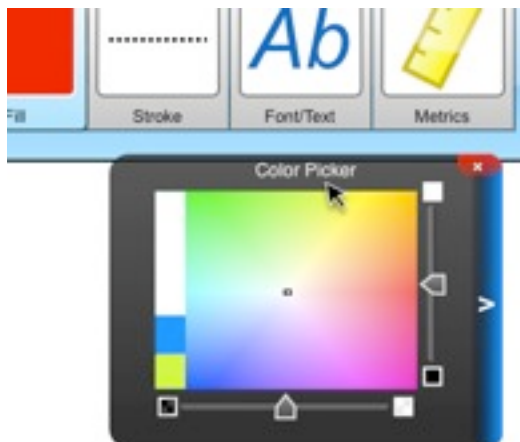


Window Behavior:

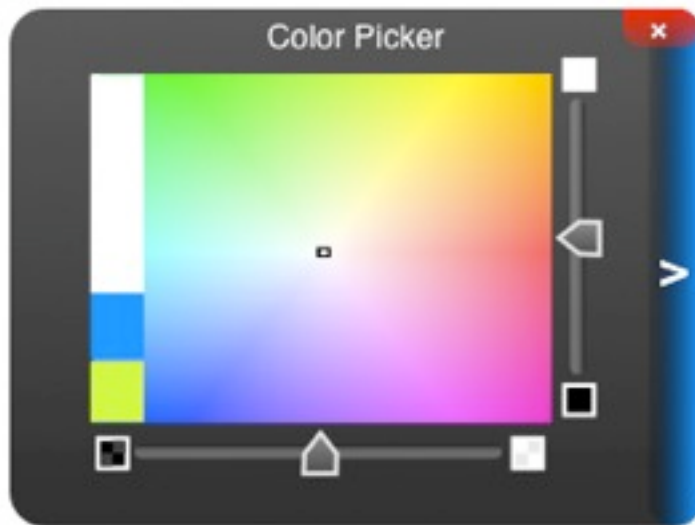
The window will remain visible until (a. The user moves their mouse away for more than 2sec. (It would be nice to have a “fade” animation wherein the window dissolves into the background), (b. The user clicks on the “X” in the top-right corner, or (c. The user makes a “final” decision in the window. In the example of the color picker, the user could close it by double clicking the color-select area to signify that they had completed their work.. Also, by dragging the bottom-right corner, the window could be expanded.

De-Anchoring

Some people may want a certain window to always be visible, for quick access. Again, there are two ways to “De-anchor” the window from the command. Either Clicking on the “+” sign at the upper right, or clicking and holding on the “fabric” of the window (The semi-transparent part), i.e., dragging it away from its location. A de-anchored color picker would look like this (Below). The “De-anchor” button is now gone. There is now a title for the window, so as to avoid confusion when there are multiple de-anchored windows. This de-anchored window can be dragged anywhere about the workspace. As you can see, the original command is still glowing, to show that it is still being actively modified. It will remain this way until the window is closed. Now, I am going to explain how the color picker itself works:



Color Picker:

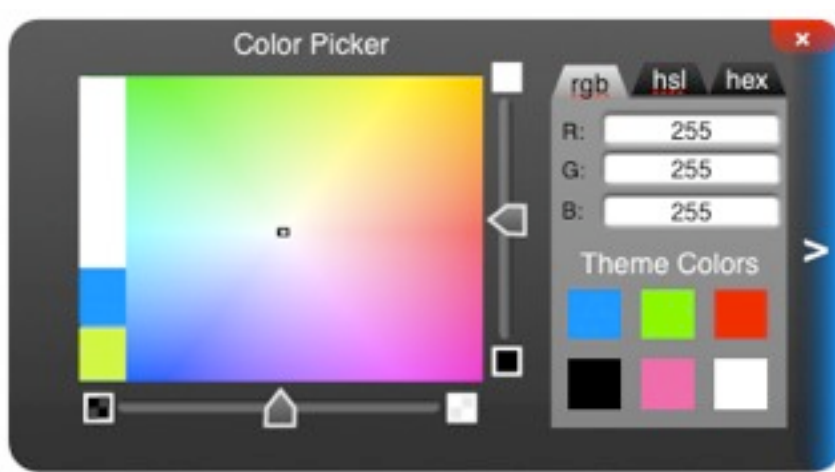


The color picker (Below), is a far cry from the current (clunky) system. In this new version, the user simply clicks once on the area in the hue selector (currently a screenshot from Apple's color picker). If we actually go through with this, we will create another hue selector. The slider on the right changes the brightness/darkness of the color. The slider at the bottom sets the alpha of the color, from the left (Completely transparent), to the right (Opaque). The mockup is not accurate in this regard.

On the left of the color picker, you can see three large colored areas. The largest is the currently selected color (The result of the hue, value, and opacity settings), and the two smaller ones are

the most recent colors.

Now, let me explain the large blue bar on the right. The bar opens the extra options for the window. In this case, the result of clicking it might look like:



There are now options for setting/viewing RGB, HSL, and Hexadecimal color values. Also, there is a grid showing color suggestions from the current theme

I have developed mockups for an improved font selector menu, as well as a stroke-selector, but I would like to hear feedback on the current idea.

I appreciate your input! Even “This will never work, it will hog resources and take years to code” is input!